Package 'aNCA'

December 9, 2025

Title (Pre-)Clinical NCA in a Dynamic Shiny App

Version 0.1.0

Description An interactive 'shiny' application for performing non-compartmental analysis (NCA) on pre-clinical and clinical pharmacokinetic data. The package builds on 'PKNCA' for core estimators and provides interactive visualizations, CDISC
outputs ('ADNCA', 'PP', 'ADPP') and configurable TLGs (tables, listings, and graphs). Typical use cases include exploratory
analysis, validation, reporting or teaching/demonstration of NCA methods. Methods and core estimators are described in Denney, Duvvuri, and Buckeridge (2015) ``Simple, Automatic Noncompartmental Analysis: The PKNCA R Package'' <doi:10.1007/s10928-015-9432-2>.

License Apache License (>= 2)

Imports dplyr, formatters, flextable, ggplot2, magrittr, methods, officer, PKNCA (>= 0.12.1), plotly, purrr, rlang, rlistings, scales, stats, stringr, tern, tidyr, units, utils

Encoding UTF-8

RoxygenNote 7.3.3

Suggests arrow, covr, DT, ggh4x, glue, haven, htmltools, htmlwidgets, jsonlite, knitr, lintr (>= 3.1.2), logger, markdown, mockery, nestcolor, openxlsx2, pak, reactable, reactable.extras, rmarkdown, sass, shiny, shinycssloaders, shinyjs, shinyjqui, shinytest2, shinyWidgets, stringi, testthat (>= 3.0.0), tools, quarto, vdiffr, withr, yaml

VignetteBuilder knitr

Config/testthat/edition 3

Language en-US

URL https://pharmaverse.github.io/aNCA/, https://github.com/pharmaverse/aNCA

BugReports https://github.com/pharmaverse/aNCA/issues

Depends R (>= 4.1)

LazyData true

NeedsCompilation no

Author Ercan Suekuer [aut] (ORCID: <https://orcid.org/0009-0001-1626-1526>),
Gerardo Jose Rodriguez [aut] (ORCID:
<https://orcid.org/0000-0003-1413-0060>),
Pascal Baertschi [aut] (ORCID: <https://orcid.org/0000-0002-6533-0399>),
Jana Spinner [aut] (ORCID: <https://orcid.org/0009-0009-2197-9530>),
Mateusz Kolomanski [aut] (ORCID:
<https://orcid.org/0000-0001-7424-3919>),
Lucy Aspridis [aut] (ORCID: <https://orcid.org/0009-0003-1300-3622>),
Valentin Legras [aut, cre],
F. Hoffmann-La Roche AG [cph, fnd]

Maintainer Valentin Legras <anca.pharmaverse@gmail.com>

Repository CRAN

Date/Publication 2025-12-09 07:40:12 UTC

Contents

`.apply_slope_rules` *Apply Slope Rules to Update Data*

Description

This function iterates over the given slopes and updates the `data$conc$data` object by setting inclusion or exclusion flags based on the slope conditions.

Usage

```
.apply_slope_rules(data, slopes, slope_groups)
```

Arguments

`data`	A list containing concentration data (`data$conc$data`) with columns that need to be updated based on the slope rules.
`slopes`	A data frame containing slope rules, including `TYPE`, `RANGE`, and `REASON` columns. May also have grouping columns (expected to match slope_groups)
`slope_groups`	A character vector specifying the group columns used for filtering.

Value

description The modified `data` object with updated inclusion/exclusion flags and reasons in `data$conc$data`.

`.compress_range` *Compresses a numeric vector into the simplest possible character string that, when evaluated, will create the same numeric vector.*

Description

Compresses a numeric vector into the simplest possible character string that, when evaluated, will create the same numeric vector.

Usage

```
.compress_range(range_vector)
```

Arguments

`range_vector`	numeric vector with numbers to compress into string

Value

simplest possible character string representing provided vector

.eval_range *Evaluates range notation. If provided notation is invalid, returns NA.*

Description

Evaluates range notation. If provided notation is invalid, returns NA.

Usage

```
.eval_range(x)
```

Arguments

x	character string with range notation, e.g. 1:5.

Value

numeric vector with specified range of numbers, NA if notation is invalid

.plotly_empty_plot *Generate an Empty Plotly Object*

Description

This function returns a blank Plotly plot with optional annotation text. It ensures that when no valid data is available, a meaningful placeholder plot is displayed instead of causing an error.

Usage

```
.plotly_empty_plot(message = "No data available")
```

Arguments

message	A character string specifying the text to display in the center of the empty plot. Defaults to "No data available".

Value

A Plotly object representing an empty plot with hidden axes.

add_f_to_pknca_results *Add bioavailability results to PKNCA results*

Description

Add bioavailability results to PKNCA results

Usage

```
add_f_to_pknca_results(res_nca, f_aucs)
```

Arguments

res_nca	A list containing non-compartmental analysis (NCA) results, including concentration and dose data.
f_aucs	A character vector of the comparing AUC parameter/s including the prefix f_ (e.g., c("f_aucinf.obs", "f_auclast")).

Value

A PKNCA result object which results data frame contains added the bioavailability calculations requested based on the AUCs provided in f_aucs.

add_optional_layers *Helper function to handle optional layers*

Description

Helper function to handle optional layers

Usage

```
add_optional_layers(
  plt,
  yaxis_scale,
  show_threshold,
  threshold_value,
  show_dose,
  data,
  time_scale,
  facet_by = NULL
)
```

Arguments

plt	The ggplot object to modify
yaxis_scale	The scale of the y-axis ("Log" or "Linear")
show_threshold	Whether to show a threshold line
threshold_value	The value of the threshold line
show_dose	Whether to show dose times as vertical lines
data	The data used for plotting
time_scale	The time scale used for plotting
facet_by	Variables to facet the plot by #' @returns The modified ggplot object with optional layers added

---

add_pptx_sl_plot *Add a slide with a plot only*

Description

Add a slide with a plot only

Usage

```
add_pptx_sl_plot(pptx, plot)
```

Arguments

pptx	rpptx object
plot	ggplot object to show as plot

Value

rpptx object with slide added

add_pptx_sl_plottable *Add a slide with both a plot and a table*

Description

Add a slide with both a plot and a table

Usage

```
add_pptx_sl_plottable(pptx, df, plot)
```

Arguments

pptx	rpptx object
df	Data frame to show as table
plot	ggplot object to show as plot

Value

rpptx object with slide added

add_pptx_sl_table *Add a slide with a table only*

Description

Add a slide with a table only

Usage

```
add_pptx_sl_table(
  pptx,
  df,
  title = "",
  footer = "Click here for individual results"
)
```

Arguments

pptx	rpptx object
df	Data frame to show as table
title	Title text for the slide
footer	Footer text for the slide

Value

rpptx object with slide added

add_qmd_plot *Helper to create a Quarto code chunk for a plot*

Description

Used internally to format a plot chunk for Quarto documents.

Usage

```
add_qmd_plot(plot_expr, use_plotly = FALSE)
```

Arguments

plot_expr	Expression for plot.
use_plotly	Logical, whether to convert plot to plotly.

Value

Character vector for Quarto code chunk.

add_qmd_sl_plot *Add a slide to the Quarto document with a single plot*

Description

Used internally for dose escalation reporting.

Usage

```
add_qmd_sl_plot(quarto_path, plot, use_plotly = FALSE)
```

Arguments

quarto_path	Path to the Quarto (.qmd) file to append to.
plot	Expression for plot.
use_plotly	Logical, whether to convert plot to plotly.

Value

Invisibly returns TRUE if the slide was added.

add_qmd_sl_plottabletable

Add a slide to the Quarto document with a plot and two tables (side by side)

Description

Used internally for dose escalation reporting.

Usage

```
add_qmd_sl_plottabletable(quarto_path, df1, df2, plot, use_plotly = FALSE)
```

Arguments

quarto_path	Path to the Quarto (.qmd) file to append to.
df1	Expression for first table (left column).
df2	Expression for second table (right column).
plot	Expression for plot.
use_plotly	Logical, whether to convert plot to plotly.

Value

Invisibly returns TRUE if the slide was added.

add_qmd_table

Helper to create a Quarto code chunk for a table

Description

Used internally to format a table chunk for Quarto documents.

Usage

```
add_qmd_table(table_expr)
```

Arguments

table_expr	Expression for table.

Value

Character vector for Quarto code chunk.

`apply_filters` *Apply Filters to a Dataset*

Description

This function applies a set of filters to a dataset. Each filter specifies a column, condition, and value to filter the dataset.

Usage

```
apply_filters(data, filters)
```

Arguments

`data`	A data frame containing the raw data to be filtered.
`filters`	A list of filters, where each filter is a list containing the column, condition, and value.

Details

The function iterates over the list of filters and applies each filter to the dataset. The supported conditions are ==, >, <, >=, <= and !=.

Value

A data frame containing the filtered data.

Examples

```
# Example usage:
data <- mtcars
filters <- list(
  list(column = "mpg", condition = ">", value = "20"),
  list(column = "cyl", condition = "==", value = "6")
)
filtered_data <- apply_filters(data, filters)
```

apply_labels *Apply Labels to a dataset*

Description

This function adds "label" attributes to all columns in a dataset

Usage

```
apply_labels(data, labels_df = metadata_nca_variables, type = "ADNCA")
```

Arguments

data	The dataset to which labels will be applied.
labels_df	A data frame containing at least the columns "Variable", "Label", and "Dataset".
type	The type variable in labels_df for which the labels are to be applied.

Value

The same dataset with label attributes applied to all columns. If a column is not present in the labels list, it will be assigned the name of the col. If label already exists in the original data, it will be preserved.

Examples

```
data <- data.frame(USUBJID = c(1, 2, 3), AVAL = c(4, 5, 6))
labels <- data.frame(
  Variable = c("USUBJID", "AVAL"),
  Label = c("Unique Subject Identifier", "Analysis Value"),
  Dataset = c("ADNCA", "ADNCA")
)
data <- apply_labels(data, labels, "ADNCA")
print(attr(data$USUBJID, "label")) # "Unique Subject Identifier"
print(attr(data$AVAL, "label"))    # "Analysis Value"
```

apply_mapping *Apply UI-Based Column Mapping to a Dataset*

Description

This function takes a dataset and applies user-specified column mappings provided through a Shiny input object. It renames columns based on the selected mappings, handles special cases such as ADOSEDUR, updates units for key variables, applies labels, and removes detected duplicate concentration records.

Usage

```
apply_mapping(
  dataset,
  mapping,
  desired_order,
  silent = TRUE,
 req_mappings = c("USUBJID", "AFRLT", "NFRLT", "ARRLT", "NRRLT", "PCSPEC", "ROUTE",
    "AVAL", "STUDYID", "ATPTREF", "AVALU", "RRLTU", "DOSEU", "PARAM")
)
```

Arguments

dataset	A data frame containing the raw data to be transformed.
mapping	A named list of column mappings.
desired_order	A character vector specifying the desired column order in the output dataset.
silent	Boolean, whether to print message with applied mapping. Defaults to TRUE.
req_mappings	A character vector indicating the names of the mapping object that must always be populated

Details

- Validates that all required columns are mapped and no duplicates exist.
- If ADOSEDUR is not mapped, it is assigned a value of 0.
- Removes concentration data duplicates using all columns except ARRLT, NRRLT, and ATPTREF.

Value

A transformed data frame with:

- Renamed columns according to user mappings
- Created columns for mapped non-existent variables (e.g., mg/L in AVALU)
- Created columns for unmapped required variables (e.g., ADOSEDUR)
- Ordered columns as specified
- Labels applied to columns (via apply_labels())
- Concentration duplicates removed based on key identifiers: AFRLT, STUDYID, PCSPEC, DOSETRT, USUBJID, and PARAM

calculate_f *Calculate bioavailability with pivoted output*

Description

This function calculates bioavailability (F) based on AUC (Area Under Curve) data extracted from `res_nca`. It computes individual bioavailability where IV and EX data are available for a subject. If IV data is missing, it estimates bioavailability using the mean IV values for that grouping. The output is pivoted such that each row represents all main results summarized for each profile in each subject. Columns are assumed to be in `%` units even if not explicitly stated.

Usage

```
calculate_f(res_nca, f_aucs)
```

Arguments

`res_nca`	A list containing non-compartmental analysis (NCA) results, including concentration and dose data.
`f_aucs`	A character vector of the comparing AUC parameter/s including the prefix f_ (e.g., `c("f_aucinf.obs", "f_auclast")`).

Value

A pivoted data frame with bioavailability calculations (`f_aucinf`, `f_auclast`, etc.) for individual subjects where IV data is available. If IV data is missing for the subject, the mean IV AUC for that group is used instead. Variables are assumed to be in `%` units.

calculate_ratios *Calculate Ratios from PKNCA Results*

Description

Calculate Ratios from PKNCA Results

Usage

```
calculate_ratios(
  data,
  test_parameter,
  ref_parameter = test_parameter,
  match_cols,
  ref_groups,
  test_groups = NULL,
  adjusting_factor = 1,
  custom_pptestcd = NULL
)
```

Arguments

data	A PKNCAresults object or its result data.frame.
test_parameter	Character. The PPTESTCD value to use as test (numerator).
ref_parameter	Character. The PPTESTCD value to use as reference (denominator). Defaults to test_parameter.
match_cols	Character vector of column names to match between test and reference groups or a data.frame specifying columns and values.
ref_groups	A data.frame specifying reference groups. At its minimum, contains the contrast variable value(s) for the reference.
test_groups	A data.frame specifying test groups. Optional. By default is NULL, allowing rows not in ref_groups be used as test.
adjusting_factor	Numeric to multiply the ratio. Default is 1.
custom_pptestcd	Optional character. If provided, will be used as the PPTESTCD value.

Value

A data.frame result object with the calculated ratios.

---

calculate_summary_stats

Calculate Summary Statistics

Description

This function calculates various summary statistics for formatted output of PKNCA::pk.nca().

Usage

```
calculate_summary_stats(data, input_groups = "ATPTREF")
```

Arguments

data	A data frame containing results of Non Compartmental Analysis using PKNCA package. Assumes presence of columns: PPORRES, PPSTRES, PPSTRESU, PPTESTCD
input_groups	A character vector specifying the columns to group by. Here. the hierarchical order matters Default is "PPSTRESU".

Details

The function calculates the following statistics for numeric variables:

- Geometric mean (`geomean`)
- Geometric coefficient of variation (`geocv`)
- Arithmetic mean (`mean`)
- Standard deviation (`sd`)
- Minimum value (`min`)
- Maximum value (`max`)
- Median value (`median`)
- Count of missing values (`count.missing`)
- Count (`count`)

The resulting summary statistics are rounded to three decimal places. If units are different, they are standardized to the group's most frequent first unit.

Value

A data frame with summary statistics for each group and parameter.

Examples

```
data <- data.frame(
  ATPTREF = c(1, 1, 1, 1, 1, 1),
  PPTESTCD = c("A", "A", "B", "B", "C", "C"),
  PPORRES = c(10, 20, 5, 15, NA, 30),
  PPSTRES = c(10, 20, 5, 15, NA, 30),
  PPORRESU = c("mg/L", "mg/L", "ng/mL", "ng/mL", "µg/L", "µg/L"),
  PPSTRESU = c("mg/L", "mg/L", "ng/mL", "ng/mL", "µg/L", "µg/L")
)
calculate_summary_stats(data)
```

---

check_slope_rule_overlap

Check overlap between existing and new slope rulesets

Description

Takes in tables with existing and incoming selections and exclusions, finds any overlap and differences, edits the ruleset table accordingly.

Usage

```
check_slope_rule_overlap(existing, new, slope_groups, .keep = FALSE)
```

Arguments

existing	Data frame with existing selections and exclusions.
new	Data frame with new rule to be added or removed.
slope_groups	List with column names that define the groups.
.keep	Whether to force keep fully overlapping rulesets. If FALSE, it will be assumed that the user wants to remove rule if new range already exists in the dataset. If TRUE, in that case full range will be kept.

Value

Data frame with full ruleset, adjusted for new rules.

---

convert_volume_units *Convert Volume Units to Match Concentration Denominator Units*

Description

This function identifies rows associated with excretion samples (e.g., urine, feces, bile) and adjusts the VOLUME and VOLUMEU columns so that the volume unit matches the denominator unit in the corresponding concentration unit (AVALU). This is necessary for PKNCA calculation of excretion parameters.

Usage

```
convert_volume_units(
  df,
  avalu = "AVALU",
  volume = "VOLUME",
  volumeu = "VOLUMEU"
)
```

Arguments

df	A data frame containing pharmacokinetic data.
avalu	A character string specifying the column name for concentration values (default: "AVALU").
volume	A character string specifying the column name for volume or mass values (default: "VOLUME").
volumeu	A character string specifying the column name for volume or mass units (default: "VOLUMEU"). It must contain the following columns: **PCSPEC** Sample type (e.g., urine, feces, bile, plasma). **AVAL** Concentration values. **AVALU** Concentration units (e.g., "ug/mL", "mg/g"). **VOLUME** Volume or mass values for integration. **VOLUMEU** Units for the VOLUME column (e.g., "mL", "g").

Details

It uses the **units** package to perform unit-safe conversions. If a direct conversion between volume and the concentration denominator is not possible (e.g., between mass and volume), a fallback conversion is attempted using a neutral density of 1 (`target_unit / original_unit`). The function modifies only the VOLUME and VOLUMEU columns when necessary and leaves all other data unchanged.

The function:

1. Parses the denominator from AVALU (e.g., "ug/mL" → "mL").
2. Attempts to convert the corresponding VOLUME to that unit.
3. If direct conversion fails, assumes a neutral density of 1 (i.e., 1 `unit_target / unit_original`) and retries.
4. Leaves units unchanged for non-excreta samples or already-valid combinations.

The function assumes that the AVALU column contains concentration units in the form of "x/y" (e.g., "ug/mL", "mg/g").

Value

A modified data frame with VOLUME and VOLUMEU converted (where necessary) so that multiplying AVAL * VOLUME results in a unit with consistent dimensionality (typically mass or moles). A new column AMOUNTU is created to represent the product of AVALU and VOLUMEU.

Examples

```
df <- data.frame(
  PCSPEC = c("urine", "feces", "plasma"),
  AVAL = c(100, 5, 70),
  AVALU = c("ug/mL", "mg/g", "ng/mL"),
  VOLUME = c(2, 1.5, 3),
  VOLUMEU = c("L", "mL", "mL"),
  stringsAsFactors = FALSE
)

df_converted <- convert_volume_units(df)
```

---

create_html_dose_slides

Render dose escalation results to HTML via Quarto

Description

Used internally to create and render a .qmd file to HTML.

Usage

```
create_html_dose_slides(res_dose_slides, path, title)
```

Arguments

res_dose_slides	List of results for each dose group.
path	Path to the output HTML file.
title	Title for the presentation.

Value

Invisibly returns TRUE if rendering succeeded.

---

create_metabfl *Create METABFL Column Based on Selected Metabolites*

Description

Create METABFL Column Based on Selected Metabolites

Usage

```
create_metabfl(dataset, metabolites)
```

Arguments

dataset	A data frame containing a dataset with a PARAM (parameter) column.
metabolites	A character vector of parameter names representing metabolites.

Value

The input dataset with an additional METABFL column indicating metabolite records ("Y") or non-metabolite records ("").

---

create_pptx_doc *Create a new PowerPoint document from a template and add a title slide*

Description

Create a new PowerPoint document from a template and add a title slide

Usage

```
create_pptx_doc(path, title, template)
```

Arguments

path	File path to save the presentation
title	Title for the presentation
template	Path to PowerPoint template file

Value

rpptx object

---

create_pptx_dose_slides

Create a PowerPoint presentation with dose escalation results, including main and extra figures Adds slides for summary tables, mean plots, line plots, and individual subject results

---

Description

Create a PowerPoint presentation with dose escalation results, including main and extra figures Adds slides for summary tables, mean plots, line plots, and individual subject results

Usage

```
create_pptx_dose_slides(res_dose_slides, path, title, template)
```

Arguments

res_dose_slides	List of results for each dose group
path	File path to save the presentation
title	Title for the presentation
template	Path to PowerPoint template file

Value

TRUE (invisible). Writes the PowerPoint file to the specified path

create_qmd_doc *Create a new Quarto presentation file with YAML header and setup chunk*

Description

Used internally to initialize a Quarto document for exporting plots and tables.

Usage

```
create_qmd_doc(
  quarto_path,
  title = "NCA Report",
  libraries = c("plotly", "flextable", "dplyr"),
  rda_path = NULL,
  template = NULL,
  extra_setup = NULL
)
```

Arguments

quarto_path	Path to the Quarto (.qmd) file to create.
title	Title for the presentation.
libraries	Character vector of libraries to load in setup chunk.
rda_path	Path to the RDS file to be loaded in the document.
template	(Optional) Path to a Quarto template to use (default: NULL).
extra_setup	(Optional) Character vector of extra setup lines to include after YAML.

Value

Invisibly returns TRUE if the file was created.

create_qmd_dose_slides *Create all slides for dose escalation results in a Quarto document*

Description

Used internally to generate main and individual slides for each dose group.

Usage

```
create_qmd_dose_slides(res_dose_slides, quarto_path, title, use_plotly = TRUE)
```

Arguments

res_dose_slides	List of results for each dose group.
quarto_path	Path to the Quarto (.qmd) file to create.
title	Title for the presentation.
use_plotly	Logical, whether to convert plots to plotly.

Value

Invisibly returns TRUE if slides were created.

---

create_start_impute *Create C0 Impute Column*

Description

Defines an impute column in the intervals of the PKNCAdata object based on data

Usage

```
create_start_impute(pknca_data)
```

Arguments

pknca_data	A PKNCAdata object containing concentration and dose data.

Value

A PKNCAdata object with updated intervals table including start imputation strategies.

Examples

```
adnca <- read.csv(system.file("shiny/data/Dummy_data.csv", package = "aNCA"))
pknca_data <- PKNCA_create_data_object(adnca)
pknca_data <- create_start_impute(pknca_data)
```

`detect_study_types` *Detect study types*

Description

This function detects the type of study based on the provided data.

Usage

```
detect_study_types(
  data,
  groups,
  metabfl_column,
  route_column,
  volume_column = "volume"
)
```

Arguments

`data`	The dataset containing the study types to be identified. Assumed to be the output from aNCA formatted concentration data. Must contain group columns, the specified route, analyte and drug columns, and ADOSEDUR.
`groups`	the grouping variables for the study type detection.
`metabfl_column`	A character string specifying the column name for the metabolite flag, which is coded as "Y" for metabolites
`route_column`	A character string specifying the column name for the route of administration.
`volume_column`	A character string specifying the column name for the volume of a sample. Extravascular samples must be written as `extravascular`. Can be set to `volume` if not applicable.

Details

The function identifies a possible five different types of studies based on grouping by `STUDYID`, `DOSETRT`, `USUBJID`, `PCSPEC`, and the route column. The study types are determined as follows:

- "Excretion Data": If the volume column for a group is not NA and has values > 0.
- "Single Extravascular Dose": If there is only one dose and TRTRINT is not available, and the route is extravascular.
- "Single IV Infusion Dose": If there is only one dose and TRTRINT is not present, and the route is not extravascular.
- "Single IV Bolus Dose": If there is only one dose and TRTRINT is not present, the route is not extravascular, and ADOSEDUR == 0.
- "Multiple Extravascular Doses": If there are multiple doses (or TRTRINT is available) and the route is extravascular.

- "Multiple IV Infusion Doses": If there are multiple doses (or TRTRINT is available) and the route is not extravascular.
- "Multiple IV Bolus Doses": If there are multiple doses or TRTRINT is not present, the route is not extravascular, and ADOSEDUR == 0.
- If none of these conditions are met, the type is marker as "Unknown".

Value

A data frame summarizing the detected study types, including the grouping columns and the identified type.

Examples

```
sample_data <- data.frame(
  STUDYID = "STUDY001",
  DOSETRT = "Drug",
  ANALYTE = "DOSETRT",
  USUBJID = c(
    # 1. Single IV Dose subject
    "Subj-01", "Subj-01",
    # 2. Multiple Extravascular Doses subject (identified by TRTRINT)
    "Subj-02", "Subj-02",
    # 3. Excretion Data subject (identified by positive volume)
    "Subj-03", "Subj-03"
  ),
  PCSPEC = "PLASMA",
  DOSNOA = c(
    1, 1,        # Single dose
    1, 1,        # Appears as single dose...
    1, 1         # Single dose
  ),
  ROUTE = c(
    "INTRAVENOUS", "INTRAVENOUS",
    "extravascular", "extravascular",
    "INTRAVENOUS", "INTRAVENOUS"
  ),
  ADOSEDUR = c(
  0, 0,
  0, 0,
  2, 2),
  SAMPLE_VOLUME = c(
    NA, 0,
    NA, 0,
    10, 12       # Positive volume indicates excretion
  ),
  TRTRINT = c(
    NA, NA,
    24, 24,      # ...but TRTRINT indicates a multiple-dose regimen
    NA, NA
  ),
  METABFL = c(
    "N", "N",
```

```
    "N", "N",
    "Y", "Y"    # mark last subject as metabolite
  )
)

study_summary <- detect_study_types(
  data = sample_data,
  groups = c("USUBJID", "PCSPEC", "DOSETRT"),
  metabfl_column = "METABFL",
  route_column = "ROUTE",
  volume_column = "SAMPLE_VOLUME"
)
```

`dose_profile_duplicates` *Create duplicates in concentration data with Pre-dose and Last Values for Dosing Cycles*

Description

This function duplicates and adjusts concentration data to ensure all dosing cycles have complete pre-dose and last concentration values. It is designed for use in pharmacokinetic analyses where dosing intervals and concentration values need to be aligned for each dose.

Usage

```
dose_profile_duplicates(
  conc_data,
  groups = c("USUBJID", "DOSNOA", "PARAM"),
  dosno = "DOSNOA",
  arrlt = "ARRLT",
  afrlt = "AFRLT",
  nrrlt = "NRRLT",
  nfrlt = "NFRLT"
)
```

Arguments

`conc_data`	A data frame containing concentration data.
`groups`	A character vector of column names to use for grouping (e.g., c("USUBJID", "PARAM", "DOSNOA")).
`dosno`	Column name for the dose number (default: "DOSNOA").
`arrlt`	Column name for time from the most recent dose.
`afrlt`	Column name for time from the first dose.
`nrrlt`	Column name for the numeric relative time.
`nfrlt`	Column name for the nominal relative time.

Value

A data frame with adjusted concentration data, including:

- Duplicated pre-dose values assigned to the previous dose.
- Duplicated last values assigned to the next dose if pre-dose values are missing, filtered so only samples within 4 hours of the next dose are included.
- Sorted by the grouping variables and relative time.

Examples

```
# Example usage
conc_data <- data.frame(
USUBJID = c("001", "001", "001", "001", "001", "001", "001", "001", "001", "001"),
AVAL = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
DOSNOA = c(1, 1, 1, 2, 2, 2, 2, 3, 3, 3),
ARRLT = c(-1, 0, 1, -1, 0, 1, 2, 0, 1, 2),
AFRLT = c(-1, 0, 1, 2, 3, 4, 5, 6, 7, 8),
NRRLT = c(-1, 0, 1, -1, 0, 1, 2, 0, 1, 2),
NFRLT = c(-1, 0, 1, 2, 3, 4, 5, 6, 7, 8)
)
result <- dose_profile_duplicates(conc_data,
                              groups = c("USUBJID", "DOSNOA"), dosno = "DOSNOA")
```

---

ensure_column_unit_exists

Ensure Unit Columns Exist in PKNCA Object

Description

Checks if specified unit columns exist in a PKNCA object (either PKNCAconc or PKNCAdose). If the columns do not exist, it creates them and assigns default values (NA or existing units).

Usage

```
ensure_column_unit_exists(pknca_obj, unit_name)
```

Arguments

pknca_obj	A PKNCA object (either PKNCAconc or PKNCAdose).
unit_name	A character vector of unit column names to ensure (concu, amountu, timeu...).

Details

The function performs the following steps:

1. Checks if the specified unit columns exist in the PKNCA object.
2. If a column does not exist, it creates the column and assigns default values.
3. If not default values are provided, it assigns NA to the new column.

Value

The updated PKNCA object with ensured unit columns.

`export_cdisc` *Export CDISC Data*

Description

This function processes the results from a PKNCA and exports them into CDISC compliant datasets. Attention: All parameters that do no match pptest dataframe will be lost in this pipeline!

Usage

```
export_cdisc(res_nca)
```

Arguments

`res_nca`	Object with results of the NCA analysis.

Details

Outputs are the following:

- pknca_result Output from function call `pk.nca()` (formatted)
- pknca_result_raw Output from function call `pk.nca()` (needs to be merged with upper later on but now we avoid merge conflict)

Value

A list with two data frames:

pp A data frame containing the PP (Pharmacokinetic Parameters) domain data.

adpp A data frame containing the ADPP (Analysis Dataset for Pharmacokinetic Parameters) domain data.

`filter_breaks` *Filter Breaks for X-Axis*

Description

Filters X-axis for consecutive breaks with at least the specified distance.

Usage

```
filter_breaks(breaks = NA, plot = plot, min_cm_distance = 0.5, axis = "x")
```

Arguments

`breaks`	A numeric vector of x-axis breaks.
`plot`	A ggplot object used to extract plot dimensions and scales.
`min_cm_distance`	A numeric of the minimum distance between breaks.
`axis`	Axis to filter on, either "x" or "y".

Value

A numeric vector of filtered x-axis breaks.

Author(s)

Gerardo Rodriguez

`filter_slopes` *Filter dataset based on slope selections and exclusions*

Description

This function filters main dataset based on provided slope selections an exclusions.

Usage

```
filter_slopes(data, slopes, profiles, slope_groups, check_reasons = FALSE)
```

Arguments

data	Data to filter. Must be PKNCAdata list, containing the conc element with PKNCAconc list and appropriate data frame included under data.
slopes	A data frame containing slope rules, including TYPE, RANGE, and REASON columns. May also have grouping columns (expected to match slope_groups)
profiles	List with available profiles for each SUBJECT.
slope_groups	List with column names that define the groups.
check_reasons	Whether to check if all selections have REASONS stated. If this is TRUE and not all selections have a reason provided, an error will be thrown.

Value

Original dataset, with is.included.hl, is.excluded.hl and exclude_half.life columns modified in accordance to the provided slope filters.

---

flexible_violinboxplot

Flexible Violin/Box Plot

Description

This function generates a violin or box plot based on the provided data, parameter, and dose information.

Usage

```
flexible_violinboxplot(
  res_nca,
  parameter,
  xvars,
  colorvars,
  varvalstofilter = NULL,
  columns_to_hover,
  box = TRUE,
  plotly = TRUE,
  seed = NULL
)
```

Arguments

res_nca	A PKNCA results object containing the results and concentration data.
parameter	A string specifying the parameter to be plotted.
xvars	Variables for the x axis.
colorvars	Variables for the color aesthetic.

varvalstofilter
: Character vector specifying which variable and value to pre-filter as `colname: value`. By default is NULL (no pre-filtering)

columns_to_hover
: A character vector indicating the column names from result_data that should be used to identify when hovering the plotly outputs

box
: A logical value indicating whether to plot a box plot (`TRUE`) or a violin plot (`FALSE`). Default is `TRUE`.

plotly
: A logical value defining if the output is plotly (TRUE, default) or ggplot otherwise (FALSE)

seed
: An integer value to set the seed for reproducibility of jittering. Default (NULL) will use the current R seed.

Value

A plotly object representing the violin or box plot.

---

format_pkncaconc_data *Create PK Concentration Dataset*

Description

This function creates a pharmacokinetic concentration dataset from the provided ADNCA data.

Usage

```
format_pkncaconc_data(
  ADNCA,
  group_columns,
  time_column = "AFRLT",
  rrlt_column = "ARRLT",
  route_column = "ROUTE"
)
```

Arguments

ADNCA
: A data frame containing the ADNCA data.

group_columns
: A character vector specifying the columns to group by.

time_column
: A character string specifying the time column.

rrlt_column
: A character string specifying the time since last dose column.

route_column
: A character string specifying the route column.

Details

The function performs the following steps:

- Checks for required columns and data.
- Filters out rows with EVID = 0 and PARAMCD containing "DOSE" (dosing data- not CDISC standard)
- Creates DOSNOA variable, sequential numbers based on time of dose
- Creates a 'std_route' column with PKNCA values "intravascular" or "extravascular" based on route_column (ROUTE, CDISC: C66729).
- Arranges the data by group_columns.

Value

A data frame containing the filtered and processed concentration data.

Examples

```
adnca <- read.csv(system.file("shiny/data/Dummy_data.csv", package = "aNCA"))
conc_data <- format_pkncaconc_data(ADNCA = adnca,
                              group_columns = c("STUDYID", "DOSETRT", "USUBJID", "PARAM"),
                                    time_column = "AFRLT",
                                    rrlt_column = "ARRLT",
                                    route_column = "ROUTE")
```

---

format_pkncadata_intervals

Create Dose Intervals Dataset

Description

This function creates a dataset with dose intervals and specified pharmacokinetic parameters.

Usage

```
format_pkncadata_intervals(
  pknca_conc,
  pknca_dose,
  params = c("aucinf.obs", "aucint.last", "auclast", "cmax", "half.life", "tmax",
   "lambda.z", "lambda.z.n.points", "r.squared", "adj.r.squared", "lambda.z.time.first",
    "aucpext.obs", "aucpext.pred", "clast.obs", "cl.obs"),
  start_from_last_dose = TRUE
)
```

Arguments

pknca_conc	A PKNCAdose object containing the concentration data.
pknca_dose	A PKNCAdose object containing the dose data.
params	A character vector specifying the pharmacokinetic parameters to include.
start_from_last_dose	Logical defining if start is at time of last dose or C1.

Details

The function performs the following steps:

- Creates a vector with all pharmacokinetic parameters.
- Based on dose times, creates a data frame with start and end times.
- If TRTRINT column is present in data, sets last dose end time to start + TRTRINT, or if TRTRINT is NA then either Inf if only one dose present, or max end time if not.
- If no TRTRINT column in data, sets last dose end time to the time of last sample or Inf if single dose data.
- Adds logical columns for each specified parameter.

Assumes that multiple dose data will have a TRTRINT column or contain multiple doses in dataset

Value

A data frame containing the dose intervals and specified pharmacokinetic parameters.

Examples

```
adnca <- read.csv(system.file("shiny/data/Dummy_data.csv", package = "aNCA"))
pknca_data <- PKNCA_create_data_object(adnca)
pknca_conc <- pknca_data$conc
pknca_dose <- pknca_data$dose
params <- c("aucinf.obs", "cmax", "half.life", "tmax", "lambda.z")
dose_intervals <- format_pkncadata_intervals(pknca_conc, pknca_dose, params)
```

---

format_pkncadose_data *Create PK Dose Dataset*

Description

This function creates a pharmacokinetic dose dataset from the provided concentration data.

Usage

```
format_pkncadose_data(
  pkncaconc_data,
  time_column = "AFRLT",
  rrlt_column = "ARRLT",
  group_columns
)
```

Arguments

pkncaconc_data	A data frame containing the concentration data.
time_column	A character string specifying the time from first dose column.
rrlt_column	A character string specifying the time since last dose column.
group_columns	A character vector specifying the columns to group by.

Details

The function performs the following steps:

- Arranges and groups the data by group_columns
- Selects the first row within each group (arranged by DOSNOA- a variable created in `format_pkncaconc_data`)

Note*: This function is designed to work with the output of `format_pkncaconc_data`.

Value

A data frame containing the dose data.

---

format_unit_string *Formats a unit string if a unique unit exists*

Description

Formats a unit string if a unique unit exists

Usage

```
format_unit_string(data, unit_var)
```

Arguments

data	The data frame to check.
unit_var	The column name of the unit variable.

Value

A formatted string like " (hr)" or an empty string "".

`general_lineplot` *Generate a General Line Plot for ADNCA Dataset*

Description

This function generates a line plot for an ADNCA dataset based on user-selected analytes, subjects, and other parameters. The plot can be customized to display data on a linear or logarithmic scale and can be filtered by cycle.

Usage

```
general_lineplot(
  data,
  selected_analytes,
  selected_pcspec,
  selected_usubjids,
  colorby_var = "USUBJID",
  facet_by = NULL,
  time_scale,
  yaxis_scale,
  show_threshold = FALSE,
  threshold_value = 0,
  show_dose = FALSE,
  cycle = NULL,
  palette = NULL
)
```

Arguments

`data`	A data frame containing the ADNCA dataset.
`selected_analytes`	A character vector of selected analytes to be included in the plot.
`selected_pcspec`	A character vector of selected matrix to be included in the plot.
`selected_usubjids`	A character vector of selected unique subject identifiers (USUBJIDs) to be included in the plot.
`colorby_var`	A character string specifying the variable by which to color the lines in the plot.
`facet_by`	A character vector specifying the variables by which to facet the plot.
`time_scale`	A character string specifying the time scale. Options are "By Cycle" or other values.
`yaxis_scale`	A character string specifying the x-axis scale. Options are "Log" or other values.
`show_threshold`	A boolean specifying whether to show a threshold line or not. Default is FALSE.
`threshold_value`	A numeric value to set the y value of the threshold line. Default is 0.

show_dose	A boolean specifying whether to show dose times as vertical lines.
cycle	A character string or numeric value specifying the cycle to filter by when time_scale is "By Cycle". Default is NULL.
palette	Specification of the color palette to use for the plot.

Details

The function performs the following steps:a

- Filters the data based on the selected analytes, matrices, and subjects.
- Selects relevant columns and removes rows with missing concentration values.
- Converts 'USUBJID', 'ATPTREF', and 'DOSEA' to factors.
- Filters the data by cycle if time_scale is "By Cycle" while creating duplicates for predose samples if needed.
- Adjusts concentration values for logarithmic scale if yaxis_scale is "Log".
- Generates a line plot using the g_ipp function with the specified parameters.
- Adjusts the y-axis to logarithmic scale if yaxis_scale is "Log".
- Adds a horizontal line for the threshold value if show_threshold is TRUE.
- Adds vertical lines for dose times if show_dose is TRUE and the number of subjects is less than 5.

Value

A ggplot object representing the line plot of pharmacokinetic concentration over time.

Examples

```
adnca <- read.csv(system.file("shiny/data/Dummy_data.csv", package = "aNCA"))
# Use actual values from the dummy data for the example
selected_analytes <- head(unique(adnca$ANALYTE), 1)
selected_pcspec <- head(unique(adnca$PCSPEC), 1)
selected_usubjids <- head(unique(adnca$USUBJID), 1)
plot <- general_lineplot(
  data = adnca,
  selected_analytes = selected_analytes,
  selected_pcspec = selected_pcspec,
  selected_usubjids = selected_usubjids,
  colorby_var = "ATPTREF",
  time_scale = "By Cycle",
  yaxis_scale = "Log",
  cycle = "1",
  show_threshold = TRUE,
  threshold_value = 1
)
print(plot)
```

`general_meanplot` *Generate a Mean Concentration Plot for ADNCA Dataset*

Description

This function generates a mean concentration plot for an ADNCA dataset based on user-selected study IDs, analytes, and cycles. The plot can be customized to display data on a linear or logarithmic scale and can optionally include standard deviation error bars.

Usage

```
general_meanplot(
  data,
  selected_studyids,
  selected_analytes,
  selected_pcspecs,
  selected_cycles,
  id_variable = "DOSEA",
  groupby_var = c("STUDYID", "PARAM", "PCSPEC", "ATPTREF"),
  plot_ylog = FALSE,
  plot_sd_min = FALSE,
  plot_sd_max = FALSE,
  plot_ci = FALSE
)
```

Arguments

`data`	A data frame containing the ADNCA dataset.
`selected_studyids`	A character vector of selected study IDs to be included in the plot.
`selected_analytes`	A character vector of selected analytes to be included in the plot.
`selected_pcspecs`	A character vector of selected matrices to be included in the plot.
`selected_cycles`	A character vector or numeric vector of selected cycles to be included in the plot.
`id_variable`	A character string specifying the variable by which to color the lines in the plot. Default is "DOSEA".
`groupby_var`	A character string specifying the variable by which to group the data.
`plot_ylog`	A logical value indicating whether to use a logarithmic scale for the y-axis. Default is FALSE.
`plot_sd_min`	A logical value to add a SD error bar below the mean. Default is FALSE.
`plot_sd_max`	A logical value to add a SD error bar above the mean. Default is FALSE.
`plot_ci`	A logical value indicating whether to include confidence interval 95% ribbon. Default is FALSE.

Value

A ggplot object representing the mean concentration plot.

`generate_tooltip_text` *Generate HTML Tooltip Text*

Description

Generate HTML Tooltip Text

Usage

```
generate_tooltip_text(data, labels_df, tooltip_vars, type)
```

Arguments

`data`	A data.frame with the source data.
`labels_df`	A data.frame used by `get_label()` to find variable labels.
`tooltip_vars`	A character vector of column names to include in the tooltip.
`type`	A character string specifying the label type for `get_label()`.

Details

Creates a character vector of HTML tooltips for each row of a data frame, suitable for use with `ggplotly`. The output vector of this function should be added to original plotting data as a column, which then can be used as tooltip argument in the plotting function.

Value

A character vector of formatted HTML tooltip strings.

Examples

```
# Sample data
my_data <- data.frame(
  USUBJID = c("Subject-01", "Subject-02"),
  DOSE = c(100, 200),
  RESPONSE = c(5.4, 8.1)
  )

my_labels <- data.frame(
  Dataset = "ADNCA",
  Variable = "USUBJID",
  Label = "Unique Subject ID"
  ) # Dummy labels object

vars_to_show <- c("USUBJID", "DOSE", "RESPONSE")
```

```
# Generate the tooltip text vector
tooltips <- generate_tooltip_text(my_data, my_labels, vars_to_show, "ADNCA")
my_data$tooltip <- tooltips
```

---

get_conversion_factor *Transform Units*

Description

This function transforms a value from an initial unit to a target unit.

Usage

```
get_conversion_factor(initial_unit, target_unit)
```

Arguments

initial_unit	A character string representing the initial unit.
target_unit	A character string representing the target unit.

Value

A numeric value for the conversion factor from the initial to the target unit, or NA if the units are not convertible.

Examples

```
get_conversion_factor("meter", "kilometer")
get_conversion_factor("sec", "min")
```

---

get_label *Get the Label of a Heading*

Description

This function retrieves the label of a heading from a labels file.

Usage

```
get_label(variable, type = "ADNCA", labels_df = metadata_nca_variables)
```

Arguments

variable	The variable for which the label is to be retrieved.
type	The type of the dataset for which the label is to be retrieved.
labels_df	A data frame containing at least the columns "Variable", "Label", and "Dataset".

Value

The label of the heading if it exists in the labels file, otherwise the variable name.

Examples

```
LABELS <- data.frame(
  Variable = c("USUBJID", "AVAL"),
  Label = c("Unique Subject Identifier", "Analysis Value"),
  Dataset = c("ADNCA", "ADNCA")
)
get_label("USUBJID", "ADNCA", LABELS)  # Returns "Unique Subject Identifier"
get_label("AGE", "ADNCA", LABELS)  # Returns "AGE"
```

---

g_pkcg01_lin *Wrapper around aNCA::pkcg01() function. Calls the function with* LIN *scale argument.*

Description

Wrapper around aNCA::pkcg01() function. Calls the function with LIN scale argument.

Usage

```
g_pkcg01_lin(data, ...)
```

Arguments

data	Data to be passed into the plotting function.
...	Any other parameters to be passed into the plotting function.

Value

ggplot2 object for pkcg01.

---

g_pkcg01_log *Wrapper around aNCA::pkcg01() function. Calls the function with* LOG *scale argument.*

Description

Wrapper around aNCA::pkcg01() function. Calls the function with LOG scale argument.

Usage

```
g_pkcg01_log(data, ...)
```

Arguments

data	Data to be passed into the plotting function.
...	Any other parameters to be passed into the plotting function.

Value

ggplot2 object for pkcg01.

---

g_pkcg02_lin *Wrapper around aNCA::pkcg02() function. Calls the function with* LIN *scale argument.*

Description

Wrapper around aNCA::pkcg02() function. Calls the function with LIN scale argument.

Usage

```
g_pkcg02_lin(data, ...)
```

Arguments

data	Data to be passed into the plotting function.
...	Any other parameters to be passed into the plotting function.

Value

ggplot2 object for pkcg02.

---

g_pkcg02_log *Wrapper around aNCA::pkcg02() function. Calls the function with* LOG *scale argument.*

Description

Wrapper around aNCA::pkcg02() function. Calls the function with LOG scale argument.

Usage

```
g_pkcg02_log(data, ...)
```

Arguments

data	Data to be passed into the plotting function.
...	Any other parameters to be passed into the plotting function.

Value

ggplot2 object for pkcg02.

`interval_add_impute` *Add specified imputation methods to the intervals in a PKNCAdata or data.frame object.*

Description

Add specified imputation methods to the intervals in a PKNCAdata or data.frame object.

Usage

```
interval_add_impute(
  data,
  target_impute,
  after,
  target_params,
  target_groups,
  ...
)
```

Arguments

`data`	A PKNCAdata object containing the intervals data frame, or a data frame of intervals.
`target_impute`	A character string specifying the imputation method to be added.
`after`	Numeric value specifying the index in which the imputation will be added (optional). First is 0, last Inf. If missing, the imputation method is added at the end (Inf).
`target_params`	A character vector specifying the parameters to be targeted (optional). If missing, all TRUE in the intervals are taken.
`target_groups`	A data frame specifying the intervals to be targeted (optional). If missing, all relevant groups are considered.
`...`	arguments passed to `interval_add_impute`.

Details

If already present the target_impute method will be added substituting the existing one. All new intervals created will be added right after their original ones.

Value

A modified PKNCAdata object with specified imputation methods on the target intervals.

Examples

```
d_conc <- data.frame(
  conc = c(1, 0.6, 0.2, 0.1, 0.9, 0.4, 1.2, 0.8, 0.3, 0.2, 1.1, 0.5),
  time = rep(0:5, 2),
  ID = rep(1:2, each = 6),
  param = rep(c("Analyte1", "Analyte2"), each = 6)
)

d_dose <- data.frame(
  dose = c(100, 200),
  time = c(0, 0),
  ID = c(1, 2)
)

o_conc <- PKNCA::PKNCAconc(d_conc, conc ~ time | ID / param)
o_dose <- PKNCA::PKNCAdose(d_dose, dose ~ time | ID)

intervals <- data.frame(
  start = c(0, 0, 0),
  end = c(3, 5, Inf),
  half.life = c(TRUE, TRUE, TRUE),
  cmax = c(TRUE, TRUE, TRUE),
  impute = c("start_conc0,start_predose", "start_predose", "start_conc0"),
  param = c("Analyte1", "Analyte2", "Analyte1")
)

o_data <- PKNCA::PKNCAdata(o_conc, o_dose, intervals = intervals)

# Apply interval_add_impute function
o_data <- interval_add_impute(o_data,
                              target_impute = "start_conc0",
                              target_params = "half.life",
                              target_groups = data.frame(param = "Analyte1"))
```

`interval_remove_impute`

Remove specified imputation from the intervals in a PKNCAdata or data.frame (intervals) object.

Description

Remove specified imputation from the intervals in a PKNCAdata or data.frame (intervals) object.

Usage

```
interval_remove_impute(data, target_impute, ...)
```

Arguments

data	A PKNCAdata object containing the intervals data frame, or a data frame of intervals.
target_impute	A character string specifying the imputation method to remove.
...	arguments passed to interval_remove_impute.

Value

A modified object with the specified imputations removed from the targeted intervals.

Examples

```
d_conc <- data.frame(
  conc = c(1, 0.6, 0.2, 0.1, 0.9, 0.4, 1.2, 0.8, 0.3, 0.2, 1.1, 0.5),
  time = rep(0:5, 2),
  ID = rep(1:2, each = 6),
  param = rep(c("Analyte1", "Analyte2"), each = 6)
)

d_dose <- data.frame(
  dose = c(100, 200),
  time = c(0, 0),
  ID = c(1, 2)
)

o_conc <- PKNCA::PKNCAconc(d_conc, conc ~ time | ID / param)
o_dose <- PKNCA::PKNCAdose(d_dose, dose ~ time | ID)

intervals <- data.frame(
  start = c(0, 0, 0),
  end = c(3, 5, Inf),
  half.life = c(TRUE, FALSE, TRUE),
  cmax = c(TRUE, TRUE, TRUE),
  impute = c("start_conc0,start_predose", "start_predose", "start_conc0"),
  param = c("Analyte1", "Analyte2", "Analyte1")
)

o_data <- PKNCA::PKNCAdata(o_conc, o_dose, intervals = intervals)

# Apply interval_remove_impute function
o_data <- interval_remove_impute(data = o_data,
                                 target_impute = "start_conc0",
                                 target_params = "half.life",
                                 target_groups = data.frame(param = "Analyte1"))
```

---

lambda_slope_plot *Generate a Lambda Slope Plot*

Description

This function generates a lambda slope plot using pharmacokinetic data. It calculates relevant lambda parameters and visualizes the data points used for lambda calculation, along with a linear regression line and additional plot annotations.

Usage

```
lambda_slope_plot(
  conc_pknca_df,
  row_values,
  myres = myres,
  r2adj_threshold = 0.7,
  time_column = "AFRLT"
)
```

Arguments

`conc_pknca_df`	Data frame containing the concentration data (default is `mydata$conc$data`).
`row_values`	A list containing the values for the column_names used for filtering.
`myres`	A PKNCAresults object containing the results of the NCA analysis
`r2adj_threshold`	Numeric value representing the R-squared adjusted threshold for determining the subtitle color (default is 0.7).
`time_column`	The name of the time column in the concentration data frame. (default is "AFRLT").

Details

The function performs the following steps:

- Creates duplicates of the pre-dose and last doses of concentration data.
- Filters and arranges the input data to obtain relevant lambda calculation information.
- Identifies the data points used for lambda calculation.
- Calculates the fitness, intercept, and time span of the half-life estimate.
- Determines the subtitle color based on the R-squared adjusted value and half-life estimate.
- Generates a ggplot object with the relevant data points, linear regression line, and annotations.
- Converts the ggplot object to a plotly object for interactive visualization.

Value

A plotly object representing the lambda slope plot.

Examples

```
if (interactive()) {
  # Load a small packaged example dataset
  adnca <- read.csv(system.file("shiny/data/Dummy_data.csv", package = "aNCA"))

  # Subset to a single subject to keep the example fast
  subj1 <- unique(adnca$USUBJID)[3]
  dose1 <- unique(adnca$DOSNOP)[1]
  adnca_sub <- adnca[adnca$USUBJID == subj1 & adnca$DOSNOP == dose1, ]

  # Analysis details (minimal example)
  method <- "lin up/log down"
  params <- c("cmax", "tmax", "auclast", "aucinf.obs")
  analytes <- unique(adnca_sub$PARAM)
  dosnos <- unique(adnca_sub$ATPTREF)
  pcspecs <- unique(adnca_sub$PCSPEC)
  auc_data <- data.frame(start_auc = numeric(), end_auc = numeric())

  # Build a minimal PKNCA data object and run NCA (kept in \donttest for CRAN safety)
  pknca_data <- PKNCA_create_data_object(adnca_sub)
  pknca_data <- create_start_impute(pknca_data)
  pknca_data <- PKNCA_update_data_object(
    pknca_data,
    auc_data = auc_data,
    method = method,
    params = params,
    selected_analytes = analytes,
    selected_profile = dosnos,
    selected_pcspec = pcspecs
  )

  pknca_res <- PKNCA_calculate_nca(pknca_data)

  # Create the lambda slope plot for the example subject
  plot <- lambda_slope_plot(
    conc_pknca_df = pknca_data$conc$data,
   row_values = list(USUBJID = subj1, STUDYID = unique(adnca_sub$STUDYID)[1], DOSNOA = 1),
    myres = pknca_res,
    r2adj_threshold = 0.7
  )
  print(plot)
}
```

---

l_pkcl01 *Create PK Concentration Listing*

Description

This function creates a listing of pharmacokinetic (PK) concentration data segregating a dataset in lists that are customizable in title, footnotes, grouping/displayed variables, missing/zero values

and/or number of digits displayed.

Usage

```
l_pkcl01(
  data,
  listgroup_vars = c("PARAM", "PCSPEC", "ROUTE"),
  grouping_vars = c("TRT01A", "USUBJID", "ATPTREF"),
  displaying_vars = c("NFRLT", "AFRLT", "AVAL"),
  formatting_vars_table = NULL,
  title = paste0("Listing of PK Concentration by Treatment Group,",
    "Subject and Nominal Time, PK Population"),
  subtitle = NULL,
  footnote = "*: Subjects excluded from the summary table and mean plots"
)
```

Arguments

`data`	A data frame containing the PK concentration data.
`listgroup_vars`	A character vector specifying the variables to separate lists.
`grouping_vars`	A character vector specifying the grouping variables within each list.
`displaying_vars`	A character vector specifying the variables to display in the listing.
`formatting_vars_table`	A data frame with the formatting of each variable. See details.
`title`	A character string to parse specifying the main title for the entire listing.
`subtitle`	A character string to parse specifying the subtitle to use for each list.
`footnote`	A character string to parse specifying the footnote of the listing table.

Details

The function performs the following steps:

- Groups the data based on the specified grouping variables.
- Formats the 0 and NA values as defined by the formatting table.
- Creates a listing for each unique combination of the grouping variables.

The `formatting_vars_table` should be a data frame with the following columns:

- `var_name`: The name of the variable.
- `Label`: The label for the variable.
- `na_str`: The string to use for NA values.
- `zero_str`: The string to use for 0 values.
- `align`: The alignment for the variable (e.g., "center").
- `format_fun`: The formatting function to use ("round" or "signif").
- `digits`: The number of digits to use for numeric formatting.

Value

A list of listings, each corresponding to a unique combination of the grouping variables.

Author(s)

Gerardo Rodriguez

Examples

```
# Create a sample dataframe 'data' with the required variables
set.seed(123)
data <- data.frame(
  PARAM = rep(c("Param1", "Param2"), each = 6),
  PCSPEC = rep(c("Blood", "Urine"), each = 6),
  TRT01A = rep(c("Treatment1", "Treatment2"), each = 6),
  USUBJID = rep(c(rep(1, 3), rep(2, 3)), 2),
  NFRLT = rep(1:3, 4),
  AFRLT = rep(1:3, 4) + runif(12, 0, 0.5),
  TIMEU = "hours",
  AVAL = rep(0:2, 4) + runif(12, 0, 0.5),
  AVALU = "mg/L"
)

# Define the formatting table
formatting_vars_table <- data.frame(var_name = names(data),
                                    Label = c("Parameter", "Specimen", "Treatment Arm",
                                          "Unique Subject ID", "Norminal Time ($TIMEU)",
                                                "Actual Time ($TIMEU)", "Time Unit",
                                              "Analyte Value ($AVALU)", "Analyte Unit"),
                                     na_str = "Missing",
                                     zero_str = c(rep("0", 7), "BLQ", "0"),
                                     align = "center",
                                     format_fun = c(NA, NA, NA, NA,
                                                    "round", "round", NA, "round", NA),
                                     digits = c(NA, NA, NA, NA, 2, 2, NA, 3, NA))

# Call the l_pkcl01 function with the sample data
listing_ex <- l_pkcl01(data = data,
                       listgroup_vars = c("PARAM", "PCSPEC"),
                       grouping_vars = c("TRT01A", "USUBJID"),
                       displaying_vars = c("NFRLT", "AFRLT", "AVAL"),
                       formatting_vars_table = formatting_vars_table,
                       title = "Listing of PK Concentration",
                       subtitle = "Subjects with !PARAM: $PARAM (!PCSPEC: $PCSPEC)"
                       )
print(listing_ex)
```

`metadata_nca_parameters`
metadata_nca_parameters

Description

A dataset containing the mapping between PKNCA terms and CDISC terms. It mainly involves:

- Parameters: PPTEST, PPTESTCD

Usage

```
metadata_nca_parameters
```

Format

A data frame with 123 rows and 6 variables:

PKNCA PKNCA term

PPTESTCD CDISC term

PPTEST Official CDISC term

input_names Combination of PPTESTCD + ": " + PPTEST. Used for App inputs

FUN PKNCA function used to calculate the parameter

description PKNCA description of the term

is_cdisc_sure Logical indicating if the term is a CDISC official name

unit_type Type of unit associated with the term

TYPE Type of data associated with the parameter/term

CAT Arbitrary assigned subclass for the parameter/term

Depends PKNCA derived. Designates all directly used parameters in calculation

can_excretion Logical. Indicates if the parameter can be used in excretion analysis

can_non_excretion Logical. Indicates if the parameter can be used in non-excretion analysis

can_single_dose Logical. Indicates if the parameter can be used in single dose analysis

can_multiple_dose Logical. Indicates if the parameter can be used in multiple dose analysis

can_extravascular Logical. Indicates if the parameter can be used in extravascular analysis

Source

Generated for use in the `translate_nomenclature` function.

metadata_nca_variables

metadata_nca_variables

Description

A dataset containing pharmacokinetic variable specifications.

Usage

```
metadata_nca_variables
```

Format

A data frame with 361 rows and 14 variables:

Dataset Character. Indicates the dataset the variable belongs to (PP, ADNCA, ADPP).

Order Numeric. Variable order within its domain, based on Role, Core and Variable

Variable Character. The short name of the variable.

Label Character. A descriptive label for the variable.

Type Character. Data type of the variable (Char, Num, text, integer, float, dateTime).

Role Character. The CDISC role of the variable (e.g., Identifier, Topic, Timing...).

Core Character. Indicates the core status of the variable (Req = Required, Perm = Permissible, Exp = Expected, Cond = Conditional).

company_specific Logical. Indicates if the variable is company-specific (not CDISC).

is.core Logical. TRUE if the variable is a core variable (always needed to be present).

Length Numeric. The maximum length of the variable.

Controlled_Terms Character. Reference to controlled terminology (e.g., C85839, C66731).

is.used Logical. TRUE if the variable is meant to be included.

Values Character. Possible values (if applicable) for the variable separated by ', '.

is.mapped Logical. TRUE if the variable is mapped in ADNCA (App's input).

mapping_tooltip Character. Tooltip text for mapping guidance in the App.

mapping_section Character. Mapping section where the variable is classified in the App.

mapping_alternatives Character. Alternative column names for the variable.

mapping_order Numeric. Defines the mapped variables order in the mapped dataset

Source

Used for PP and ADPP mapping rules and checks in the export_cdisc function

`multiple_matrix_ratios`

Calculate Matrix Ratios This function calculates the ratios for a given data set, based on the shared time points for each matrix concentration sample. The user can input multiple tissues for which ratios should be calculated.

Description

The ratios are calculated as specimen1 / specimen 2.

Usage

```
multiple_matrix_ratios(
  data,
  matrix_col,
  conc_col,
  units_col,
  groups = c("NFRLT", "USUBJID"),
  spec1,
  spec2
)
```

Arguments

`data`	A data frame containing the concentration data.
`matrix_col`	A character string specifying the column name for the matrix type.
`conc_col`	A character string specifying the column name for the concentration data.
`units_col`	A character string specifying the column name for the units.
`groups`	A character vector of grouping variables to use for the analysis. Must include time column, USUBJID, and optionally, other grouping variables.
`spec1`	A character string specifying the value for the first specimen type(s) in the matrix_col.
`spec2`	A character string specifying the value for the second specimen type(s) in the matrix_col.

Value

A data frame containing the ratios.

Examples

```
data <- data.frame(
  USUBJID = c("A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A"),
  NFRLT = c(0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2),
  MATRIX = c(
```

```
    "BLOOD", "BLOOD", "BLOOD", "PLASMA", "PLASMA", "PLASMA",
    "BRAIN", "BRAIN", "BRAIN", "LIVER", "LIVER", "LIVER"
  ),
  CONC = c(10, 20, 15, 25, 30, 40, 5, 10, 8, 12, 18, 16),
  UNITS = rep("ng/mL", 12)
)
multiple_matrix_ratios(data, "MATRIX", "CONC", "UNITS", c("NFRLT", "USUBJID"), "BLOOD", "PLASMA")
```

`parse_annotation` *Parses annotations in the context of data. Special characters and syntax are substituted by actual data and/or substituted for format that is better parsed via rendering functions (e.g. plotly).*

Description

Parses annotations in the context of data. Special characters and syntax are substituted by actual data and/or substituted for format that is better parsed via rendering functions (e.g. plotly).

Usage

```
parse_annotation(data, text)
```

Arguments

`data`	Data frame containing data to reference. Should include columns and labels referenced in the text string. Referenced variables should be able to produce single unique result.
`text`	Character text to parse.

Details

- `\n` character is substituted for `<br>` tag in order to add new lines in rendered image.
- `$COLNAME` is parsed to provide unique data value from the mentioned column.
- `!COLNAME` is parsed to provide `label` attribute for a given column name. If any values are missing from the provided data, they are substituted for `ERR` string.

Value

Parsed annotation text.

pivot_wider_pknca_results *Reshape PKNCA Results*

Description

This function reshapes the structure of the results produced by the main function of the PKNCA package (pk.nca) in a way that each row represents all the main results summarized for each profile in each individual/subject. Excluding the ID variables, each column name corresponds with a calculated parameter and between brackets its corresponding units. AUC intervals, if present, are be added as additional columns.

Usage

```
pivot_wider_pknca_results(myres)
```

Arguments

myres	The output of PKNCA::pk.nca. It makes some additional assumptions: 1. CDISC denomination of actual and nominal time variables (AFRLT, ARRLT, NFRLT, NRRLT). 2. Intervals must include a column (type_interval) to differentiate between the custom AUC ranges ("manual") and main parameter calculations ("main"). 3. Includes PPSTRES and PPSTRESU variables in results dataset. 4. Columns start_dose and end_dose must express the actual start and end times of the dose, relative to the last reference dose. 5. Temporarily: CDISC denomination of PK parameters related to half-life: "LAMZNPT", "LAMZLL", "LAMZ" Used to derive LAMZNPT and LAMZMTD.

Value

A data frame which provides an easy overview on the results from the NCA in each profile/subject and how it was computed lambda (half life) and the results of the NCA parameters (cmax, AUC, AUClast)

pk.calc.volpk *Calculate the total urine volume*

Description

Calculate the total urine volume

Usage

```
pk.calc.volpk(volume)
```

Arguments

volume	The volume (or mass) of the sample

Value

The sum of urine volumes for the interval

---

pkcg01 *Generate PK Concentration-Time Profile Plots*

Description

This function generates a list of ggplots for PK concentration-time profiles.

Usage

```
pkcg01(
  adnca = data(),
  xvar = "AFRLT",
  yvar = "AVAL",
  xvar_unit = "RRLTU",
  yvar_unit = "AVALU",
  color_var = NULL,
  color_var_label = NULL,
  xbreaks_var = "NFRLT",
  xbreaks_mindist = 0.5,
  xmin = NA,
  xmax = NA,
  ymin = NA,
  ymax = NA,
  xlab = paste0("!", xvar, " [$", xvar_unit, "]"),
  ylab = paste0("!", yvar, " [$", yvar_unit, "]"),
  title = NULL,
  subtitle = NULL,
  footnote = NULL,
  plotgroup_vars = c("ROUTE", "PCSPEC", "PARAM", "USUBJID"),
 plotgroup_names = list(ROUTE = "Route", PCSPEC = "Specimen", PARAM = "Analyte", USUBJID
    = "Subject ID"),
  scale = c("LIN", "LOG", "SBS")[1],
  studyid = "STUDYID",
  trt_var = "TRT01A",
  plotly = TRUE
)
```

Arguments

adnca	A data frame containing the data.
xvar	A character string of the variable name for the x-axis.
yvar	A character string of the variable name for the y-axis.
xvar_unit	A character string of the unit for the x-axis variable.
yvar_unit	A character string of the unit for the y-axis variable.
color_var	A character string of the variable name for the color.
color_var_label	A character string of the color label.
xbreaks_var	A character string of the x-axis breaks.
xbreaks_mindist	A numeric value for xbreak_mindist.
xmin	A numeric value for the minimum x-axis limit.
xmax	A numeric value for the maximum x-axis limit.
ymin	A numeric value for the minimum y-axis limit.
ymax	A numeric value for the maximum y-axis limit.
xlab	Character for x-axis label. Defaults: xvar label & xvar_unit.
ylab	Character for y-axis label. Defaults: yvar label & yvar_unit.
title	Character for plot title.
subtitle	Character for plot subtitle.
footnote	A character string of a manual footnote for the plot.
plotgroup_vars	A character vector of the variables to group data.
plotgroup_names	A character vector of the grouping variable names.
scale	Scale for the Y axis, either "LIN" or "LOG".
studyid	A character string specifying the study ID variable.
trt_var	A character string specifying the treatment variable.
plotly	Logical indicating whether to return plotly objects. Defaults to TRUE.

Value

A list of ggplot or plotly objects for each unique group.

Author(s)

Gerardo Rodriguez magic numbers for footnote position and margin, work in app up to 4 lines NOTE: might require some fine tuning down the line, looks fine now

Examples

```
adnca <- read.csv(system.file("shiny/data/Dummy_data.csv", package = "aNCA"))
adnca <- subset(adnca, adnca$USUBJID %in% unique(adnca$USUBJID)[c(1, 2)])
attr(adnca[["AFRLT"]], "label") <- "Actual time from first dose"
attr(adnca[["AVAL"]], "label") <- "Analysis val"

plots_lin <- pkcg01(adnca = adnca, xmax = 1)
```

---

pkcg02 *Generate Combined PK Concentration-Time Profile Plot by Cohort*

Description

This function generates a list of plotly objects PK concentration-time profiles by group

Usage

```
pkcg02(
  adnca = data(),
  xvar = "AFRLT",
  yvar = "AVAL",
  xvar_unit = "RRLTU",
  yvar_unit = "AVALU",
  color_var = NULL,
  color_var_label = NULL,
  xbreaks_var = "NFRLT",
  xbreaks_mindist = 0.5,
  xmin = NA,
  xmax = NA,
  ymin = NA,
  ymax = NA,
  xlab = paste0("!", xvar, " [$", xvar_unit, "]"),
  ylab = paste0("!", yvar, " [$", yvar_unit, "]"),
  title = NULL,
  subtitle = NULL,
  footnote = NULL,
  plotgroup_vars = c("ROUTE", "PCSPEC", "PARAM", "TRT01A"),
 plotgroup_names = list(ROUTE = "Route", PCSPEC = "Specimen", PARAM = "Analyte", TRT01A
    = "Treatment"),
  scale = c("LIN", "LOG", "SBS")[1],
  studyid = "STUDYID",
  trt_var = "TRT01A",
  plotly = TRUE
)
```

Arguments

adnca	A data frame containing the data.
xvar	A character string of the variable name for the x-axis.
yvar	A character string of the variable name for the y-axis.
xvar_unit	A character string of the unit for the x-axis variable.
yvar_unit	A character string of the unit for the y-axis variable.
color_var	A character string of the variable name for the color.
color_var_label	A character string of the color label.
xbreaks_var	A character string of the x-axis breaks.
xbreaks_mindist	A numeric value for xbreak_mindist.
xmin	A numeric value for the minimum x-axis limit.
xmax	A numeric value for the maximum x-axis limit.
ymin	A numeric value for the minimum y-axis limit.
ymax	A numeric value for the maximum y-axis limit.
xlab	Character for x-axis label. Defaults: xvar label & xvar_unit.
ylab	Character for y-axis label. Defaults: yvar label & yvar_unit.
title	Character for plot title.
subtitle	Character for plot subtitle.
footnote	A character string of a manual footnote for the plot.
plotgroup_vars	A character vector of the variables to group data.
plotgroup_names	A character vector of the grouping variable names.
scale	Scale for the Y axis, either "LIN" or "LOG".
studyid	A character string specifying the study ID variable.
trt_var	A character string specifying the treatment variable.
plotly	Logical indicating whether to return plotly objects. Defaults to TRUE.

Value

A list of ggplot or plotly objects for each unique group.

Author(s)

Kezia Kobana magic numbers for footnote position and margin, work in app up to 4 lines NOTE: might require some fine tuning down the line, looks fine now

Examples

```
adnca <- read.csv(system.file("shiny/data/Dummy_data.csv", package = "aNCA"))
attr(adnca[["AFRLT"]], "label") <- "Actual time from first dose"
attr(adnca[["AVAL"]], "label") <- "Analysis value"

plots <- pkcg02(adnca)
plots_log <- pkcg02(adnca, scale = "LOG")
plots_custom <- pkcg02(adnca, xmin = 0, xmax = 48, title = "PK Profile", footnote = "Study X")
plotly::plotly_build(plots[[1]]) # View the first plot
```

`PKNCA_build_units_table`

Build Units Table for PKNCA

Description

This function generates a PKNCA units table including the potential unit segregating columns among the dose and/or concentration groups.

Usage

```
PKNCA_build_units_table(o_conc, o_dose)
```

Arguments

`o_conc`	A PKNCA concentration object (PKNCAconc).
`o_dose`	A PKNCA dose object (PKNCAdose).

Details

The function performs the following steps:

1. Ensures the unit columns (e.g., `concu`, `timeu`, `doseu`, `amountu`) exist in the inputs.
2. Joins the concentration and dose data based on their grouping columns.
3. Generates a PKNCA units table for each group, including conversion factors and custom units.
4. Returns a unique table with relevant columns for PKNCA analysis.

Value

A data frame containing the PKNCA formatted units table.

Examples

```
# Assuming `o_conc` and `o_dose` are valid PKNCA objects:
# 1) Sharing group variables in their formulas
# 2) Time units are the same within dose groups
# 3) Units are the same for subjects within the same concentration group

d_conc <- data.frame(
  subj = 1,
  analyte = rep(c("A", "B"), each = 2),
  concu = rep(c("ng/mL", "ug/mL"), each = 2),
  conc = c(0, 2, 0, 5),
  time = rep(0:1, 2),
  timeu = "h"
)
d_dose <- data.frame(
  subj = 1,
  dose = 100,
  doseu = "mg",
  time = 0,
  timeu = "h"
)
o_conc <- PKNCA::PKNCAconc(d_conc, conc ~ time | subj / analyte, concu = "concu")
o_dose <- PKNCA::PKNCAdose(d_dose, dose ~ time | subj, doseu = "doseu")
units_table <- PKNCA_build_units_table(o_conc, o_dose)
```

`pknca_calculate_f` *Calculate bioavailability for intravascular vs extravascular aucs*

Description

This function calculates bioavailability (F) based on AUC (Area Under Curve) data extracted from `res_nca`. It computes individual bioavailability where IV and EX data are available for a subject. If IV data is missing, it estimates bioavailability using the mean IV values for that grouping.

Usage

```
pknca_calculate_f(res_nca, f_aucs)
```

Arguments

`res_nca`	A list containing non-compartmental analysis (NCA) results, including concentration and dose data.
`f_aucs`	A character vector of the comparing AUC parameter/s including the prefix `f_` (e.g., `c("f_aucinf.obs", "f_auclast")`).

Details

- The function extracts AUC data from `res_nca$data$conc$data` and filters for selected AUC types.
- It separates data into intravascular (IV) and extravascular (EX) groups.
- Individual bioavailability is calculated for subjects with both IV and EX data using PKNCA function `pk.calc.f`.
- If IV data is missing for a subject, the function estimates bioavailability using mean IV values for that grouping.
- The final output includes bioavailability estimates for individual subjects and mean-based estimates.

Value

A data frame with calculated absolute bioavailability values (`FABS_`) for individual subjects where IV data is available. If IV data is missing, it estimates bioavailability using the mean IV AUC for that grouping.

`PKNCA_calculate_nca` *Calculates results for PKNCA analysis.*

Description

Calculates results for PKNCA analysis.

Usage

```
PKNCA_calculate_nca(pknca_data)
```

Arguments

`pknca_data`	Data object created using PKNCA::PKNCAdata() function.

Details

This function+ calculates results for PKNCA analysis using `PKNCA::pk.nca()`. It then joins the results with the dosing data, to create a full results data frame with the start and end times for each dose, from first and most recent dose.

Value

Results object with start and end times for each dose, from first dose and from most recent dose

Examples

```
example_data <- data.frame(
  STUDYID = rep("STUDY001", 6),
  PCSPEC = rep("Plasma", 6),
  ROUTE = rep("IV", 6),
  DOSETRT = rep("DrugA", 6),
  USUBJID = rep("SUBJ001", 6),
  ATPTREF = rep(1, 6),
  PARAM = rep("AnalyteA", 6),
  AVAL = c(0, 5, 10, 7, 3, 1),
  AVALU = rep("ng/mL", 6),
  DOSEA = rep(100, 6),
  DOSEU = rep("mg", 6),
  AFRLT = c(0, 1, 2, 3, 4, 6),
  ARRLT = c(0, 1, 2, 3, 4, 6),
  NFRLT = c(0, 1, 2, 3, 4, 6),
  ADOSEDUR = rep(0.5, 6),
  RRLTU = rep("hour", 6)
)

# Create a PKNCAdata object
pknca_data <- PKNCA_create_data_object(example_data)

# Perform NCA calculations
nca_results <- PKNCA_calculate_nca(pknca_data)
```

`PKNCA_create_data_object` *Creates a* `PKNCA::PKNCAdata` *object.*

Description

Creates a `PKNCA::PKNCAdata` object.

Usage

```
PKNCA_create_data_object(adnca_data)
```

Arguments

`adnca_data`	Data table containing ADNCA data.

Details

This function creates a standard PKNCAdata object from ADNCA data. It requires the following columns in the ADNCA data:

- STUDYID: Study identifier.

- PCSPEC: Matrix.
- ROUTE: Route of administration.
- DOSETRT: Drug identifier.
- USUBJID: Unique subject identifier.
- ATPTREF: (Non- standard column). Can be any column, used for filtering the data for NCA
- PARAM: Analyte.
- AVAL: Analysis value.
- AVALU: AVAL unit.
- DOSEA: Dose amount.
- DOSEU: Dose unit.
- AFRLT: Actual time from first dose.
- ARRLT: Actual time from reference dose.
- NFRLT: Nominal time from first dose.
- ADOSEDUR: Duration of dose.
- RRLTU: Time unit.

1. Creating pk concentration data using `format_pkncaconc_data()`.
2. Creating dosing data using `format_pkncadose_data()`.
3. Creating PKNCAconc object using `PKNCA::PKNCAconc()`. with formula `AVAL ~ AFRLT | STUDYID + PCSPEC + DOSETRT + USUBJID / PARAM`.
4. Creating PKNCAdose object using `PKNCA::PKNCAdose()`. with formula `DOSEA ~ AFRLT | STUDYID + DOSETRT + USUBJID`.
5. Creating PKNCAdata object using `PKNCA::PKNCAdata()`.
6. Updating units in PKNCAdata object so each analyte has its own unit.

Value

`PKNCAdata` object with concentration, doses, and units based on ADNCA data.

Examples

```
adnca_data <- data.frame(
STUDYID = rep("STUDY001", 6),
PCSPEC = rep("Plasma", 6),
ROUTE = rep("IV", 6),
DOSETRT = rep("DrugA", 6),
USUBJID = rep("SUBJ001", 6),
ATPTREF = rep(1, 6),
PARAM = rep("AnalyteA", 6),
AVAL = c(0, 5, 10, 7, 3, 1),
AVALU = rep("ng/mL", 6),
DOSEA = rep(100, 6),
DOSEU = rep("mg", 6),
AFRLT = c(0, 1, 2, 3, 4, 6),
```

```
ARRLT = c(0, 1, 2, 3, 4, 6),
NFRLT = c(0, 1, 2, 3, 4, 6),
ADOSEDUR = rep(0.5, 6),
RRLTU = rep("hour", 6)
)
PKNCA_create_data_object(adnca_data)
```

`PKNCA_hl_rules_exclusion` *Exclude NCA results based on user-defined rules over the half-life related parameters This function applies exclusion rules to the NCA results based on user-defined parameters.*

Description

Exclude NCA results based on user-defined rules over the half-life related parameters This function applies exclusion rules to the NCA results based on user-defined parameters.

Usage

```
PKNCA_hl_rules_exclusion(res, rules)
```

Arguments

`res`	A PKNCAresults object containing the NCA results.
`rules`	A list of exclusion rules where each rule is a named vector.

Details

The function iterates over the rules and applies the exclusion criteria to the NCA results. For any parameter that is not aucpext.obs or aucpext.pred it applies a minimum threshold, and for aucpext.obs and aucpext.pred it applies a maximum threshold.

Value

A PKNCAresults object with the exclusions applied.

PKNCA_impute_method_start_c1 *This function imputes the start concentration using the first concentration after dose*

Description

This function imputes the start concentration using the first concentration after dose

Usage

```
PKNCA_impute_method_start_c1(conc, time, start, end, ..., options = list())
```

Arguments

conc	Numeric vector of concentrations.
time	Numeric vector of times corresponding to the concentrations.
start	Numeric value indicating the start/dose time.
end	Numeric value indicating the end time.
...	Additional arguments (currently not used).
options	List of options (currently not used).

Details

This function adheres to the structure required by the PKNCA package to work with its functionalities.For more information, see the PKNCA Data Imputation Vignette.

Value

A data frame with imputed start concentration.

Examples

```
conc <- c(1, 2, 3, 4, 5)
time <- c(1, 2, 3, 4, 5)
start <- 0
end <- 4
PKNCA_impute_method_start_c1(conc, time, start, end)
```

PKNCA_impute_method_start_logslope *This function imputes the start concentration using the log slope method.*

Description

This function imputes the start concentration using the log slope method.

Usage

```
PKNCA_impute_method_start_logslope(
  conc,
  time,
  start,
  end,
  ...,
  options = list()
)
```

Arguments

conc	Numeric vector of concentrations.
time	Numeric vector of times corresponding to the concentrations.
start	Numeric value indicating the start/dose time.
end	Numeric value indicating the end time.
...	Additional arguments (currently not used).
options	List of options (currently not used).

Details

This function adheres to the structure required by the PKNCA package to work with its functionalities. For more information, see the PKNCA Data Imputation Vignette.

Value

A data frame with imputed start concentration.

Examples

```
conc <- c(5, 4, 3, 2, 1)
time <- c(1, 2, 3, 4, 5)
start <- 0
end <- 4
PKNCA_impute_method_start_logslope(conc, time, start, end)
```

PKNCA_update_data_object *Create a PKNCAdata Object for NCA or Slope Analysis*

Description

This function updates a previously prepared PKNCAdata object based on user selections for method, analyte, dose, specimen, and parameters.

Usage

```
PKNCA_update_data_object(
  adnca_data,
  auc_data,
  method,
  selected_analytes,
  selected_profile,
  selected_pcspec,
  params,
  should_impute_c0 = TRUE
)
```

Arguments

adnca_data	A reactive PKNCAdata object
auc_data	A data frame containing partial aucs added by user
method	NCA calculation method selection
selected_analytes	User selected analytes
selected_profile	User selected dose numbers/profiles
selected_pcspec	User selected specimen
params	A list of parameters for NCA calculation
should_impute_c0	Logical indicating if start values should be imputed

Details

Step 1: Update units in the PKNCAdata object ensuring unique analytes have their unique units

Step 2: Set PKNCAoptions for NCA calculation

Step 3: Format intervals using format_pkncadata_intervals()

Step 4: Apply filtering based on user selections and partial aucs

Step 5: Impute start values if requested

Note*: The function assumes that the adnca_data object has been created using the PKNCA_create_data_object() function.

Value

A fully configured PKNCAdata object.

---

pk_dose_qc_plot *Create a PK Dose Quality Control (QC) Plot*

Description

Generates a PK Dose QC plot by layering concentration data (as black shapes) and dose data (as colored points). It creates a single, unified legend for both data types and can return either a static ggplot or an interactive plotly object.

Usage

```
pk_dose_qc_plot(
  data_conc,
  data_dose = NULL,
  x_var,
  y_var,
  colour_var,
  shape_var,
  grouping_vars,
  other_tooltip_vars = NULL,
  x_var_units = NULL,
  colour_var_units = NULL,
  labels_df = metadata_nca_variables,
  title = NULL,
  show_pk_samples = TRUE,
  show_doses = TRUE,
  as_plotly = FALSE,
  height = NULL
)
```

Arguments

data_conc	A data.frame containing concentration data (e.g., PK samples).
data_dose	An optional data.frame containing dosing information.
x_var	Character. The column name to be used for the x-axis.
y_var	Character. The column name to be used for the y-axis.
colour_var	Character. The column in data_dose to map to color.
shape_var	Character. The column in data_conc to map to shape.
grouping_vars	Character vector. Column names to use for faceting.
other_tooltip_vars	Optional character vector of additional column names to include in the tooltip.

x_var_units	Character. The column name containing the units for the x-axis variable. It is expected that this column contains a single unique value.
colour_var_units	Character. The column name for the units of the colour variable in data_dose. It is expected that this column contains a single unique value.
labels_df	A data.frame used by helper functions to look up variable labels. It uses metadata_nca_variables as default
title	Character. The main title for the plot.
show_pk_samples	Logical. If TRUE, plots the concentration data.
show_doses	Logical. If TRUE, plots the dose data.
as_plotly	Logical. If TRUE, converts the final plot to an interactive plotly object.
height	Numeric. Desired height for the plot.

Details

Unless specified, the variables required as arguments are expected to be present in both data_conc and data_dose.

Value

A ggplot object or, if as_plotly = TRUE, a plotly object.

Examples

```
# Sample concentration data
conc_data <- data.frame(
  USUBJID = rep(paste0("S-", 1:2), each = 2),
  ACTUAL_TIME = c(0, 24, 0, 24),
  SAMPLE_TYPE = rep(c("PLASMA", "URINE"), 2),
  COHORT = "A",
  TIME_UNIT = "hr"
 )

# Sample dose data
dose_data <- data.frame(
  USUBJID = rep(paste0("S-", 1:2), each = 1),
  ACTUAL_TIME = c(0, 0),
  DOSE_LEVEL = c(100, 100),
  COHORT = "A",
  DOSE_UNIT = "mg"
)

# Generate the plot
pk_dose_qc_plot(
  data_conc = conc_data,
  data_dose = dose_data,
  x_var = "ACTUAL_TIME",
  y_var = "USUBJID",
```

```
  colour_var = "DOSE_LEVEL",
  shape_var = "SAMPLE_TYPE",
  grouping_vars = "COHORT",
  x_var_units = "TIME_UNIT",
  colour_var_units = "DOSE_UNIT",
  title = "Sample Dosing and PK Plot"
)
```

`prepare_plot_data` *Prepare Data for PK Dose QC Plotting*

Description

A helper function that validates, processes, and combines concentration and dose data. It creates the unified legend and faceting variables and calculates the factor levels for the plot scales.

Usage

```
prepare_plot_data(
  data_conc,
  data_dose,
  shape_var,
  colour_var,
  grouping_vars,
  labels_df,
  tooltip_vars,
  plot_conc_data,
  plot_dose_data
)
```

Arguments

`data_conc`	A data.frame of concentration data.
`data_dose`	An optional data.frame of dosing data.
`shape_var`	Character. The column name from `data_conc` for the legend.
`colour_var`	Character. The column name from `data_dose` for the legend.
`grouping_vars`	Character vector. Column names for faceting.
`labels_df`	A data.frame for label lookups.
`tooltip_vars`	Character vector of variables for the tooltip.
`plot_conc_data`	Logical flag derived from `show_pk_samples` and `data_conc`.
`plot_dose_data`	Logical flag derived from `show_doses` and `data_dose`.

Value

A `list` containing `data` (the processed tibble), `shape_levels`, and `colour_levels`.

preprocess_data_for_plot *Prepare Data for PK Lineplot*

Description

Prepare Data for PK Lineplot

Usage

```
preprocess_data_for_plot(
  data,
  selected_usubjids,
  selected_analytes,
  selected_pcspec,
  colorby_var,
  time_scale,
  yaxis_scale,
  cycle
)
```

Arguments

data	Raw data frame.
selected_usubjids, selected_analytes, selected_pcspec, cycle	Inputs for filtering.
colorby_var	The variable(s) to be used for coloring.
time_scale	String, either "By Dose Profile" or "Actual Time".
yaxis_scale	String, either "Log" or "Linear".

Value

A processed and filtered data.frame.

read_pk *Reads PK datasets from various file formats.*

Description

Reads PK datasets from various file formats.

Usage

```
read_pk(path)
```

Arguments

path	Character string with path to the dataset file.

Details

Currently supported file formats include:

- csv
- rds
- xlsx
- sas7bdat
- xpt
- parquet

Value

A data.frame object with loaded data.

Examples

```
df <- read_pk(system.file("shiny/data/Dummy_data.csv", package = "aNCA"))
```

---

run_app *Run the Shiny app*

Description

Run the Shiny app

Usage

```
run_app(datapath = NULL, ...)
```

Arguments

datapath	Full path to a single .csv or .rds data file.
...	Arguments passed to shiny::runApp()

Details

If a datapath is provided, the app will attempt to automatically load the specified dataset on startup. This is achieved by setting an internal option (options(aNCA.datapath = datapath)), which the app then reads. **This pre-loaded dataset can be overwritten; if a new file is uploaded using the widget within the app, it will replace the initial data for the current session.**

If datapath is NULL (default), the app will launch without pre-loading any data, and a file must be uploaded manually within the app.

Value

No return value, called for side effects to launch the Shiny application.

Examples

```
# Show the packaged example path (safe non-interactive snippet)
adnca_path <- system.file("shiny/data/Dummy_data.csv", package = "aNCA")
adnca_path

# To actually launch the app, run interactively:
if (interactive()) {
  run_app(datapath = adnca_path)
}
```

---

select_minimal_grouping_cols *Find Minimal Grouping Columns for Strata Reconstruction*

Description

This function identifies the smallest set of columns in a data frame whose unique combinations can reconstruct the grouping structure defined by the specified strata columns. It removes duplicate, constant, and redundant columns, then searches for the minimal combination that uniquely identifies each stratum.

Usage

```
select_minimal_grouping_cols(df, strata_cols)
```

Arguments

df	A data frame.
strata_cols	Column names in df whose unique combination defines the strata.

Value

A data frame containing the strata columns and their minimal set of grouping columns.

simplify_unit *Simplify compound unit expressions*

Description

This function takes a units object or a character string representing a unit expression and returns a simplified units using the units package simplifications.

Usage

```
simplify_unit(x, as_character = FALSE)
```

Arguments

x	A units object, character string, or vector of either to be simplified.
as_character	Logical. TRUE returns the result as a character, FALSE (default) as a unit object.

Value

A simplified units object, or a list of units objects if input is a vector.

Examples

```
# Using a units object
u <- units::set_units(1, "L*g/mg", mode = "standard")
simplify_unit(u)

# Using a character string
simplify_unit("(mg*L)/(mL)")
```

translate_terms *Translate Terms from One Nomenclature to Another*

Description

This function translates a character vector of terms from one nomenclature to another using a mapping file.

Usage

```
translate_terms(
  input_terms,
  mapping_col = "PKNCA",
  target_col = "PPTESTCD",
  metadata = metadata_nca_parameters
)
```

Arguments

input_terms	A character vector of terms to be translated.
mapping_col	Character indicating the column name in the metadata file of the input terms. Default is "PKNCA".
target_col	Character indicating the column name in the metadata file of the target terms. Default is "PPTESTCD".
metadata	Dataset used to do the mapping that contains the mapping and target columns.

Value

A character vector of translated terms. Input terms not available in mapping_col will be returned with the same value.

Examples

```
input_terms <- c("adj.r.squared", "ae", "nonexistent_term")
translate_terms(input_terms)
```

---

validate_pk *Validates data table with raw pk data.*

Description

Validates data table with raw pk data.

Usage

```
validate_pk(pk_data)
```

Arguments

pk_data	Object to check.

Details

Performs the following checks:

- If provided variable is of class `data.frame`.
- If number of rows in provided data frame is greater than 0.

Throws an error if any of the checks does not pass.

Value

Original, unchanged object. If any of the checks do not pass, throws an error.

`verify_parameters` *Conditionally Verify and Override PK Parameters Based on Sample Type*

Description

This helper function updates a PKNCA intervals data frame by verifying and overriding specific pharmacokinetic parameters depending on whether the sample is identified as excreta (e.g., urine, feces, bile). Parameters related to excretion (such as `ae`, `fe`, and those starting with `"clr."`) are selectively enabled only for excreta samples and set to `FALSE` otherwise.

Usage

```
verify_parameters(pknca_intervals, params, all_pknca_params)
```

Arguments

`pknca_intervals`	A data frame containing PKNCA interval information, including pharmacokinetic parameters and a `PCSPEC` column that describes the specimen type.
`params`	A character vector of parameter names selected by the user. Only these parameters will remain `TRUE` for excreta types.
`all_pknca_params`	A character vector of all pharmacokinetic parameters that may be present in `pknca_intervals`. These will be checked and updated accordingly.

Value

A modified version of the `pknca_intervals` data frame with appropriate parameters updated based on the specimen type.

Index